HTL Saalfelden
Systemplanung und Projektentwicklung

Projektdokumentation
2024 / 2025

Projektbezeichnung	MBot - Backend
Projektteam	Aberger Jonas, Haslinger Fabian, Hechenberger Tim
Erstellt am	06.02.2025
Letzte Änderung am	22.05.2025
Status	Fertig
Aktuelle Version	5.0

Änderungsverlauf

Nr.	Datum	Version	Geänderte Kapitel	Art der Änderung	Autor
1	06.02.2025	1.0	Initial / Sprint 1	Erstellung	Tim Hechenberger
2	26.02.2025	2.0	Sprint 2	Hinzufügen	Fabian Haslinger
3	27.03.2025	3.0	Alle	Komplette Überarbeitung + Sprint 3	Jonas Aberger Fabian Haslinger
4	22.05.2025	4.0	Sprint 4	Überarbeitung - Sprint 4	Tim Hechenberger
5	22.05.2025	5.0	Alle	Overview + Sprint 5	Jonas Aberger

Inhalt

1. Allgemeines / Projektübersicht

1.1 Projektbeschreibung

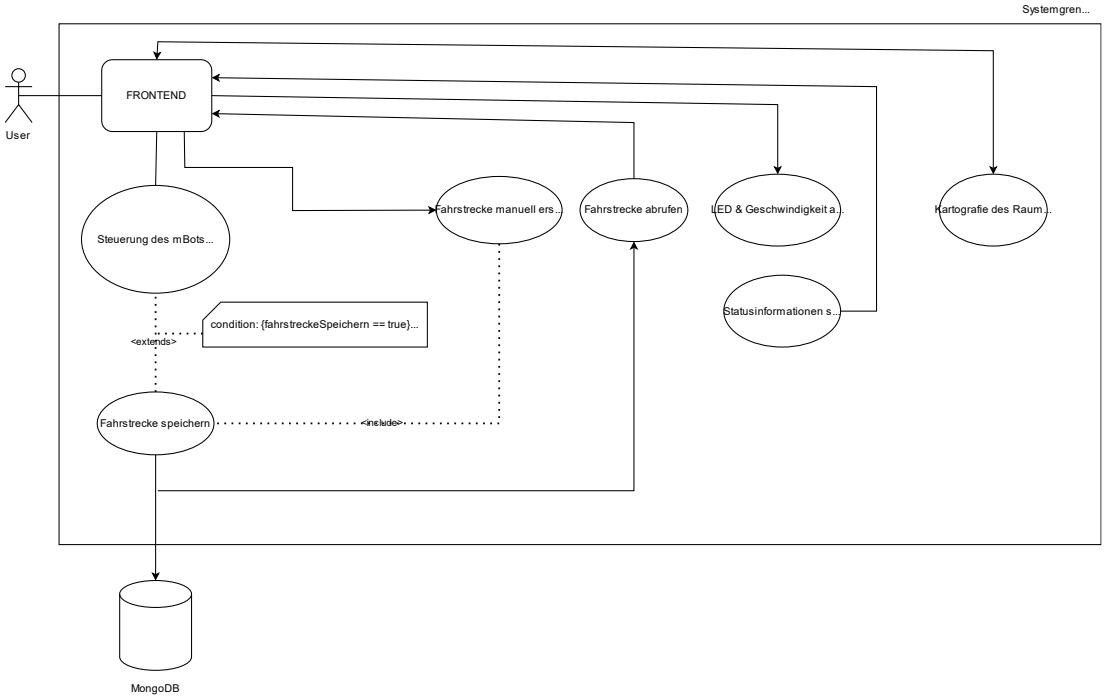
Das Projekt umfasst die Entwicklung einer Applikation zur ferngesteuerten Bedienung des MBot2-Roboters. Die Konfiguration des Roboters unterstützt eine zweiseitige Kommunikation zwischen dem Klienten und dem Endgerät, dabei wird neben dem primären Steuerungsmodul eine Vielzahl weiterer Schnittstellen implementiert.
Die Anwendung soll sowohl auf PCs als auch auf mobilen Geräten nutzbar sein.

Projektteam

Rolle(n)	Name	E-Mail	Team
Entwickler	Jonas Aberger	jonas.aberger@htl-saalfelden.at	Backend
Entwickler	Fabian Haslinger	fabian.haslinger@htl-saalfelden.at	Backend
Entwickler	Tim Hechenberger	tim.hechenberger@htl-saalfelden.at	Backend

2. Funktionale Anforderungen

2.1 Use-Case Diagramm



2.1.1 Steuerung des MBots

Der Benutzer steuert den MBot über das Frontend. Während der Steuerung gibt es die Möglichkeit das Programm mit einem Exit-Command zu beenden. Falls das Frontend die Fahrstrecke speichern will, wird der Use Case „Fahrstrecke speichern“ automatisch ausgeführt.

2.1.2 Fahrstrecke manuell erstellen

Der Benutzer kann über das Frontend eine Fahrstrecke manuell erstellen. Dies bedeutet, dass er eine Route ohne direkte Steuerung des MBot vorgibt. Dies erfolgt durch eine grafische Benutzeroberfläche, bei der vordefinierte Wegpunkte abgefahren werden. Diese Wegpunkte bestehen aus den Attributen Richtung, Geschwindigkeit und Zeit.

2.1.3 Fahrstrecke speichern

Dieser Use Case speichert die Fahrstrecke des MBot in einer **MongoDB-Datenbank**. Er kann durch zwei verschiedene Aktionen ausgeführt werden.

- Automatisch nach der Steuerung des MBot (wenn {fahrstreckenSpeichern == true}.
- Manuell durch den Use Case **„Fahrstrecke manuell erstellen“,** der das Speichern direkt einbindet´.

2.1.4 Fahrstrecke abrufen

Der Benutzer kann eine bereits gespeicherte Fahrstrecke aus der Datenbank auswählen und abfahren. Damit eine Fahrstrecke jedoch abgerufen werden kann, muss zuvor mindestens eine Strecke in der Datenbank gespeichert worden sein.

2.1.5 LED & Geschwindigkeit anpassen

Der Benutzer hat die Möglichkeit, die LEDs des MBot ein- oder auszuschalten, indem er entweder einen physischen Knopf betätigt oder eine digitale Schaltfläche in der Benutzeroberfläche auswählt. Wenn der Benutzer den Knopf drückt, werden die LEDs des MBot entweder aktiviert oder deaktiviert, abhängig von der gewählten Einstellung.

2.1.6 Statusinformationen senden

Das System tauscht kontinuierlich Daten aus, um dem Benutzer aktuelle Statusinformationen zur Hardware anzuzeigen. Diese Informationen geben einen klaren Überblick über den Zustand der Hardware, wie beispielsweise Batteriestatus oder weitere Sensordaten. Dadurch behält der Benutzer stets den Überblick und kann bei Bedarf reagieren.

2.1.7 Kartografie des Raumes

Der Benutzer möchte den umliegenden Raum erkunden und automatisch eine übersichtliche Karte erstellen lassen. Dieser Kartografie-Modus läuft eigenständig und unabhängig von anderen Funktionen. Der Benutzer kann ihn aktivieren, und der MBot beginnt, den Raum systematisch zu erfassen und auszumappen. Die gesammelten Daten werden in Echtzeit verarbeitet und als detaillierte Karte dargestellt, die dem Benutzer zur Verfügung steht.

3. Nichtfunktionale Anforderungen

Hier werden alle nichtfunktionalen Anforderungen der Backend-Gruppe-1 beschrieben:

- **Systemanforderungen:**
  - Das Backend wird als Webservice entwickelt und läuft auf einem lokalen Rechner
  - Die Kommunikation erfolgt über UDP

- **Leistung und Skalierbarkeit:**
  - Das System soll in der Lage sein, mehrere Roboter gleichzeitig zu verwalten
  - Keine zu hohen Latenzzeiten bei Steuerbefehlen und Synchronisation

- **Speicher und Ressourcennutzung:**
  - Kein zu hohem Speicherbedarf bei speichern der Daten, …

4. Projektplanung

4.1 Variantenbildung

- Basierend auf den Projektanforderungen wurden **verschiedene Varianten** analysiert:

  - Programmiersprache: Python (mit MicroPython), Java, C#
    - Gewählte Variante: Python mit MicroPython

  - Datenbank: FireBase, MongoDB, SQL
    - Gewählte Variante: MongoDB

4.2 Allgemeine Planungsinformationen

Allgemeine Planungsinformationen:

- Die Entwicklung erfolgt nach dem SCRUM-Framework, wobei das Grundkonzept, die Architektur sowie erste Tests bereits umgesetzt wurden
- Der Code und die Dokumentation werden in einem GitHub-Repository verwaltet
- Jedes Teammitglied erstellt am Anfang eines Arbeitstages einen eigenen Branch, auf welchem die Veränderungen gepusht werden.
- Am Ende der Arbeitswoche werden alle Branches zurück auf den Main – mit der Aufsicht eines zweiten Entwicklers – gemerged.

4.3 Projektumfeldanalyse

Vergleichbare Produkte

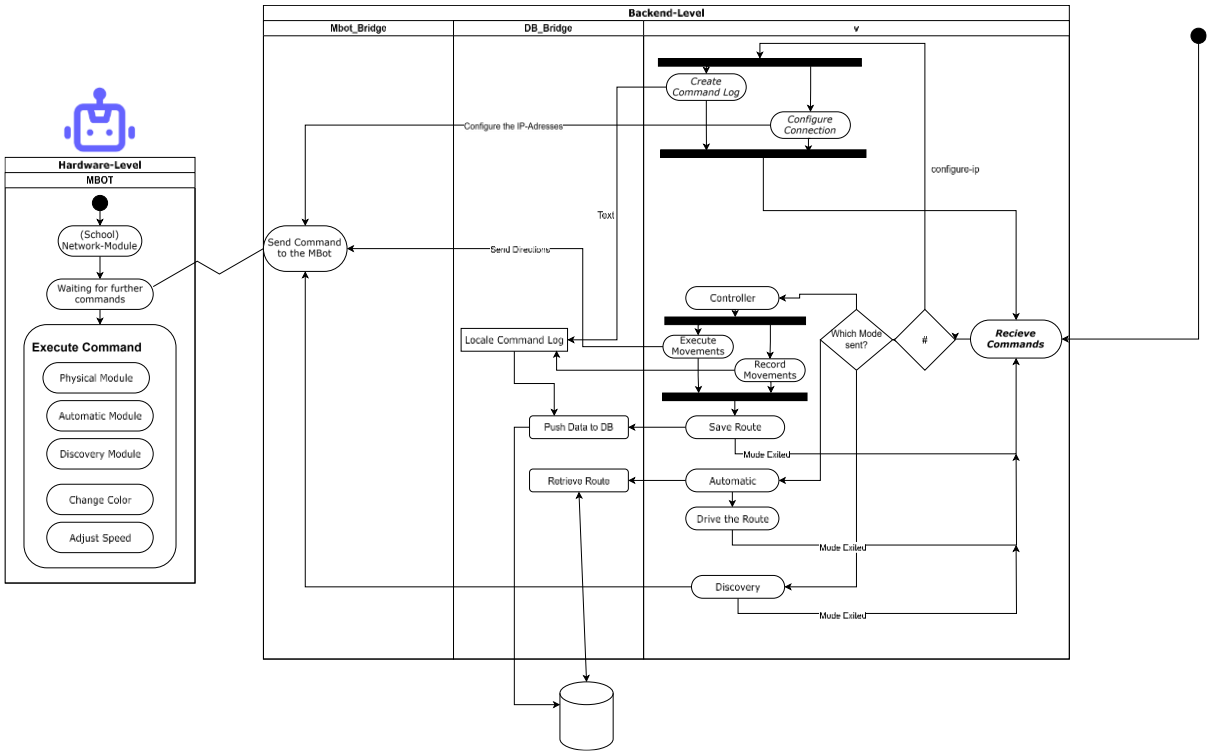
- Es gibt verschiedene MBot2-Steuerungs-Apps von Mblock, die aber keine individuelle Steuerung und Speicherung von Fahrstrecken bieten
- Bestehende Lösungen basieren meist auf Scratch, während unser System eine direkte Steuerung über Python und eine API ermöglicht

Abgrenzung

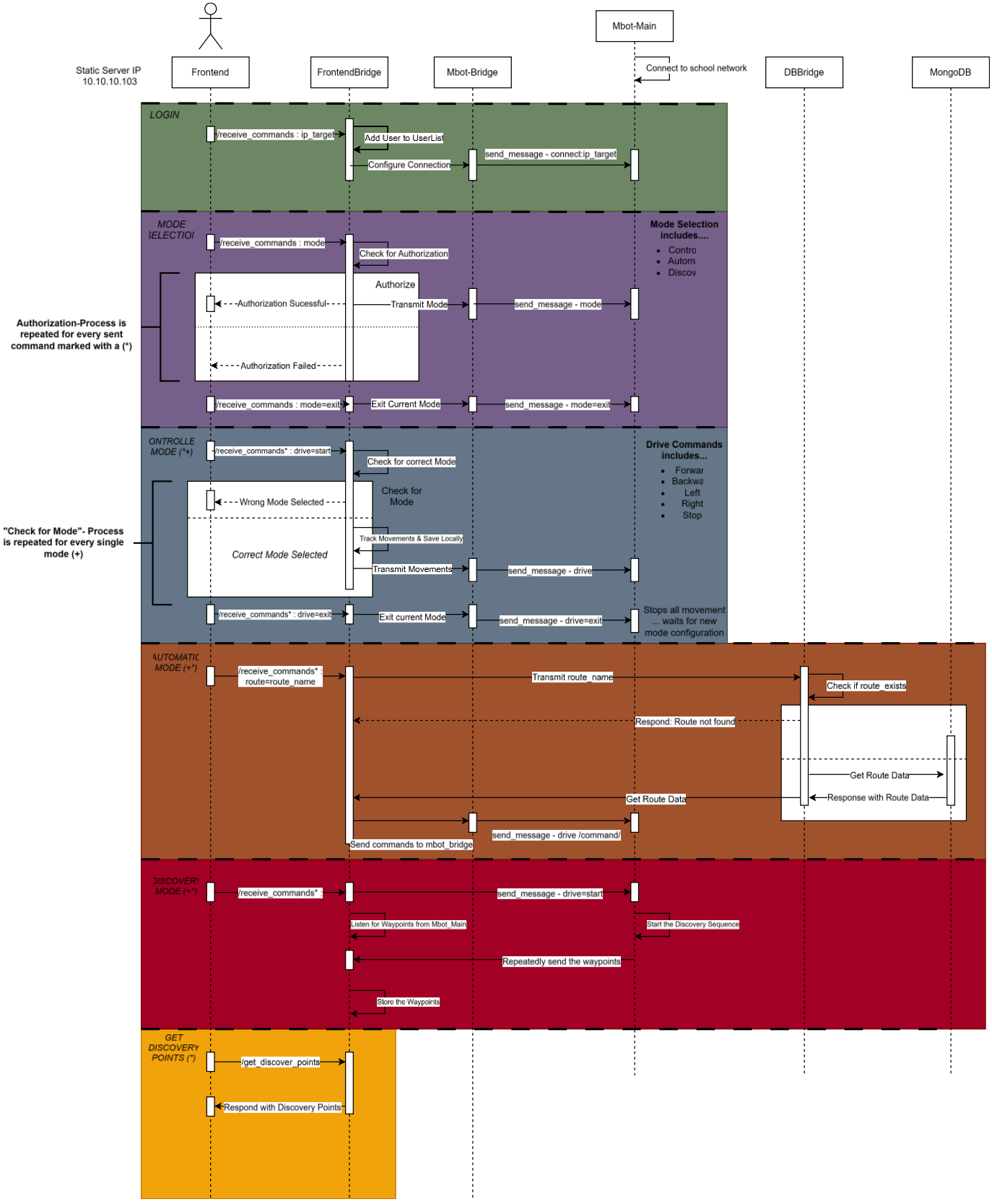
- Unser Projekt ermöglicht eine plattformunabhängige Steuerung (PC & Mobil)
- Der MBot2 kann nicht nur manuell gesteuert werden, sondern auch automatisierte Fahrstrecken abfahren
- Erweiterbare Backend-Architektur, die zukünftige Anpassungen erlaubt

5. Softwarearchitektur

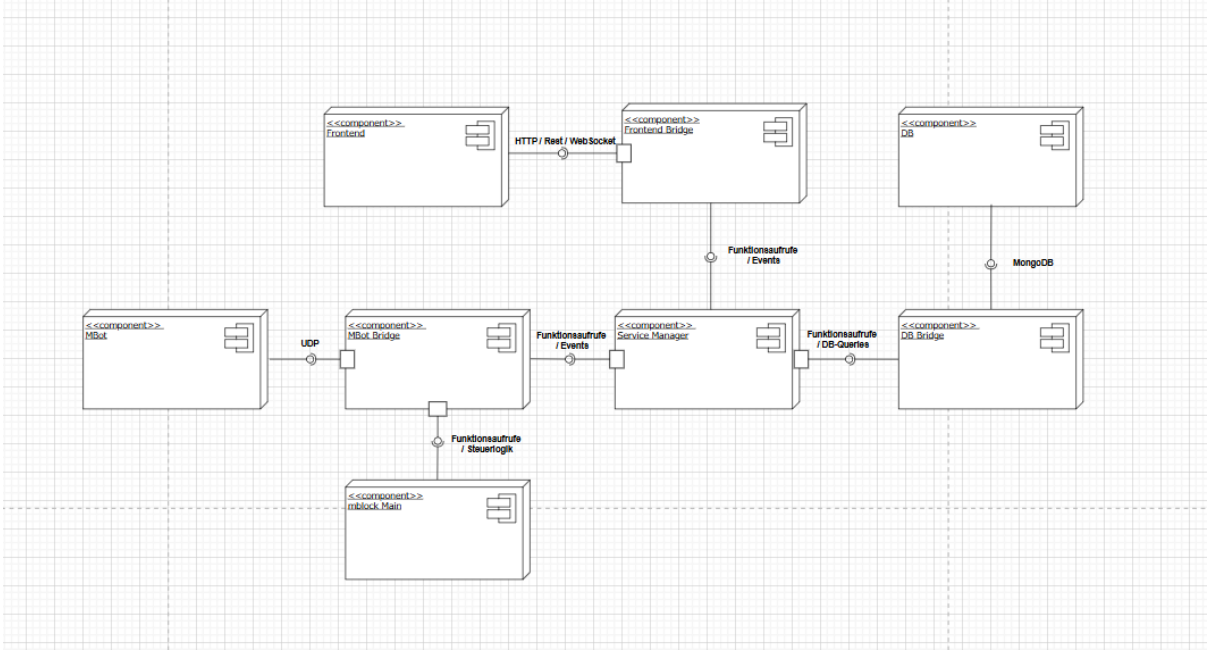
5.1 Aktivitätsdiagramm – Backend Architektur



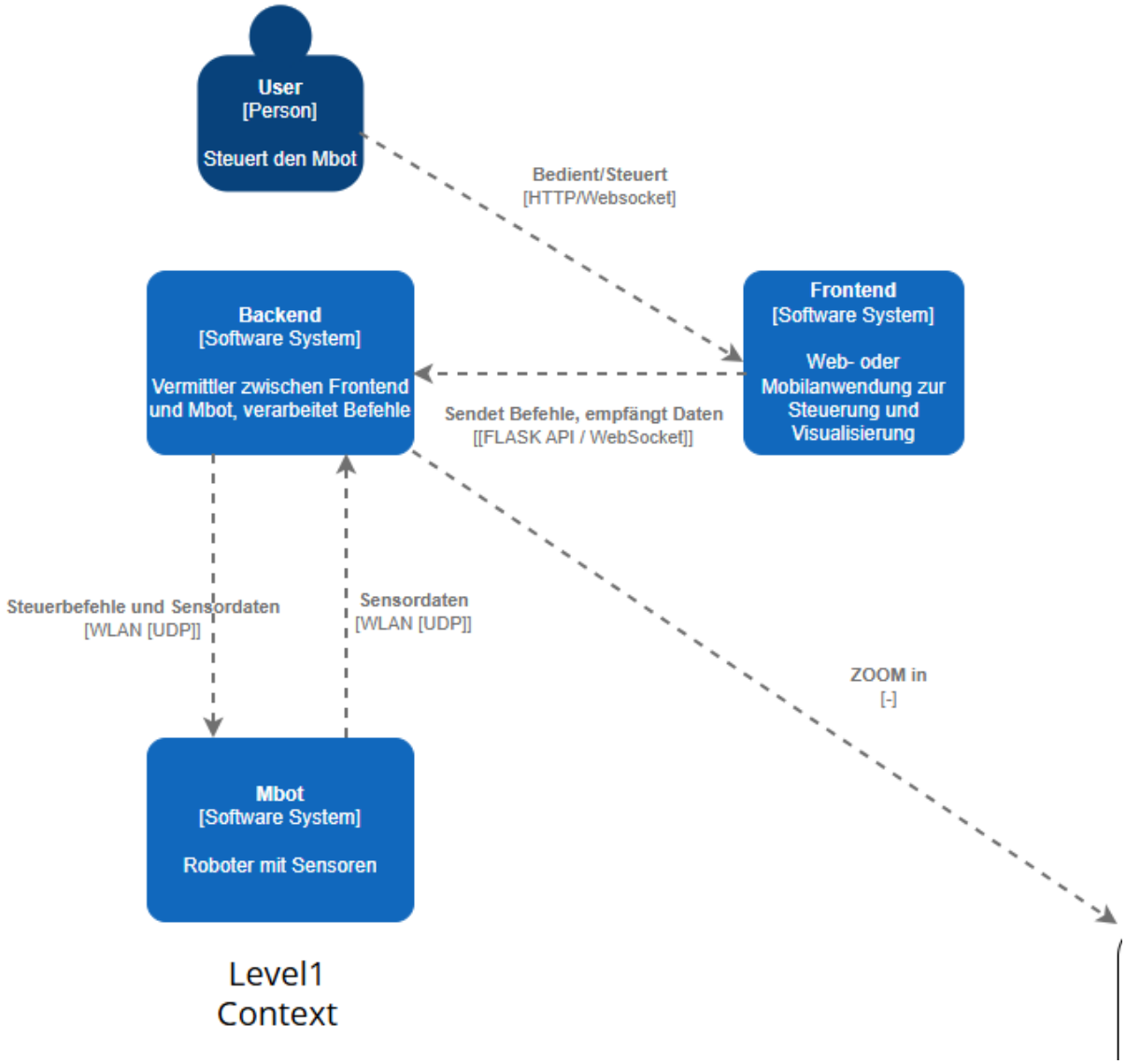
5.2 Sequenzdiagramme – Primärfunktionen Backend



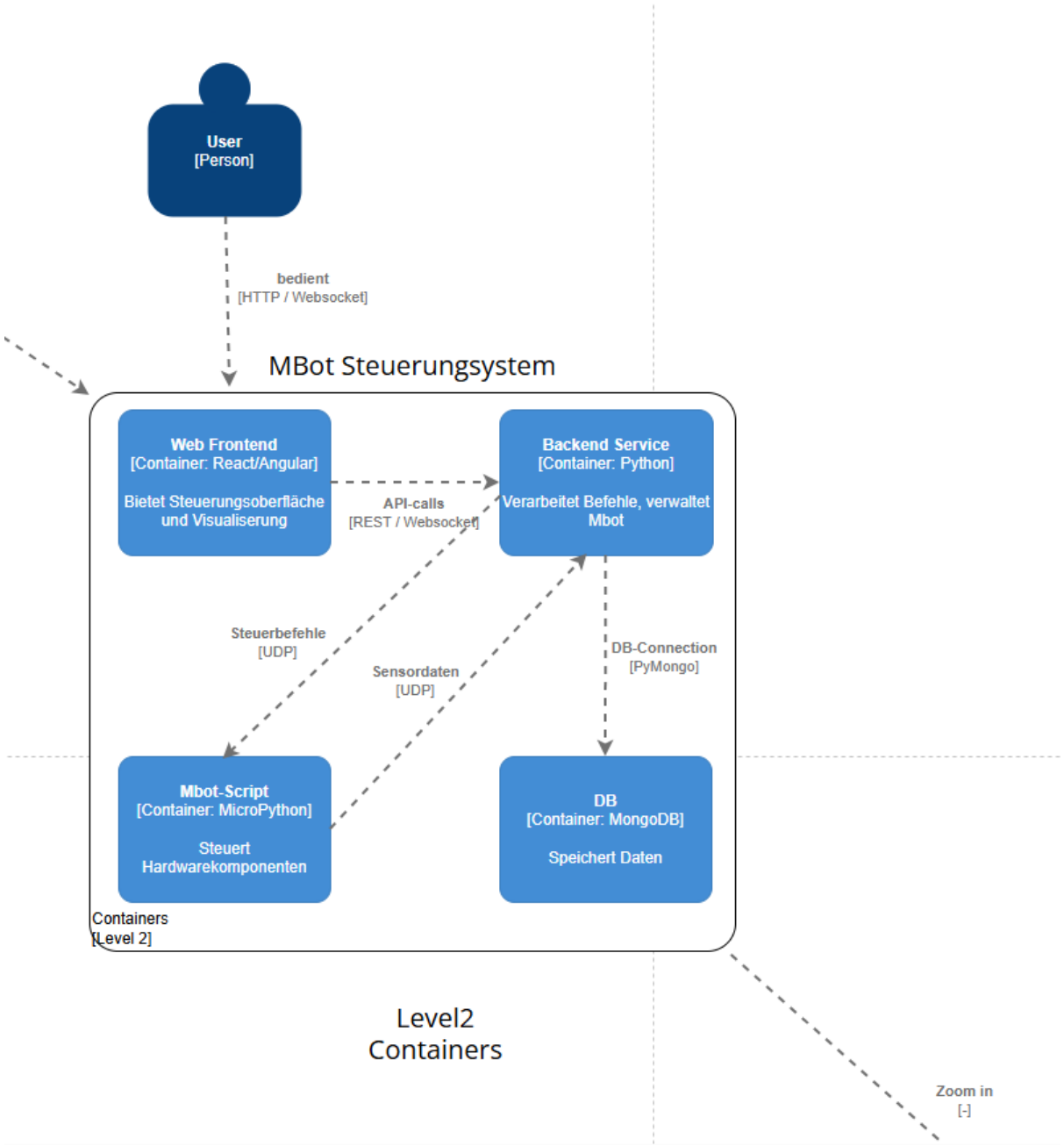
5.3 Komponentendiagramme

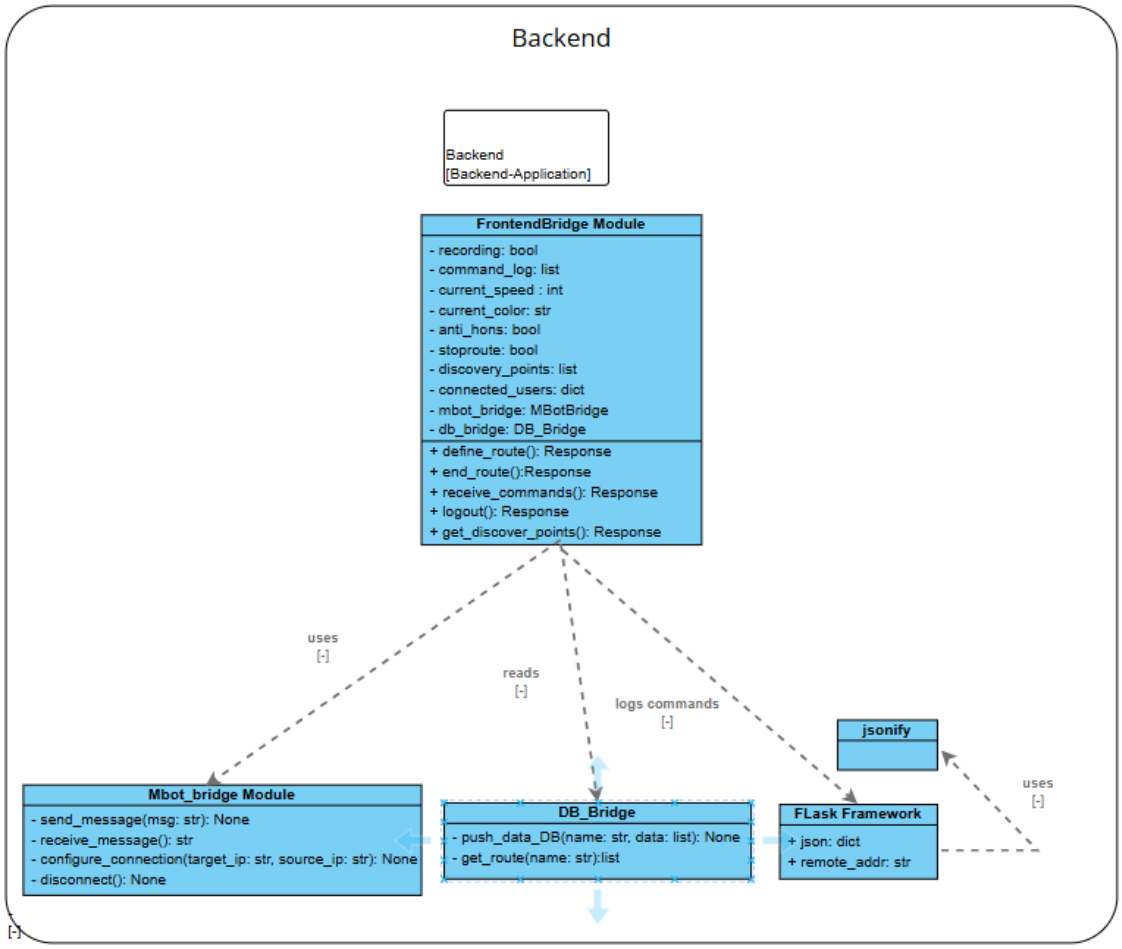


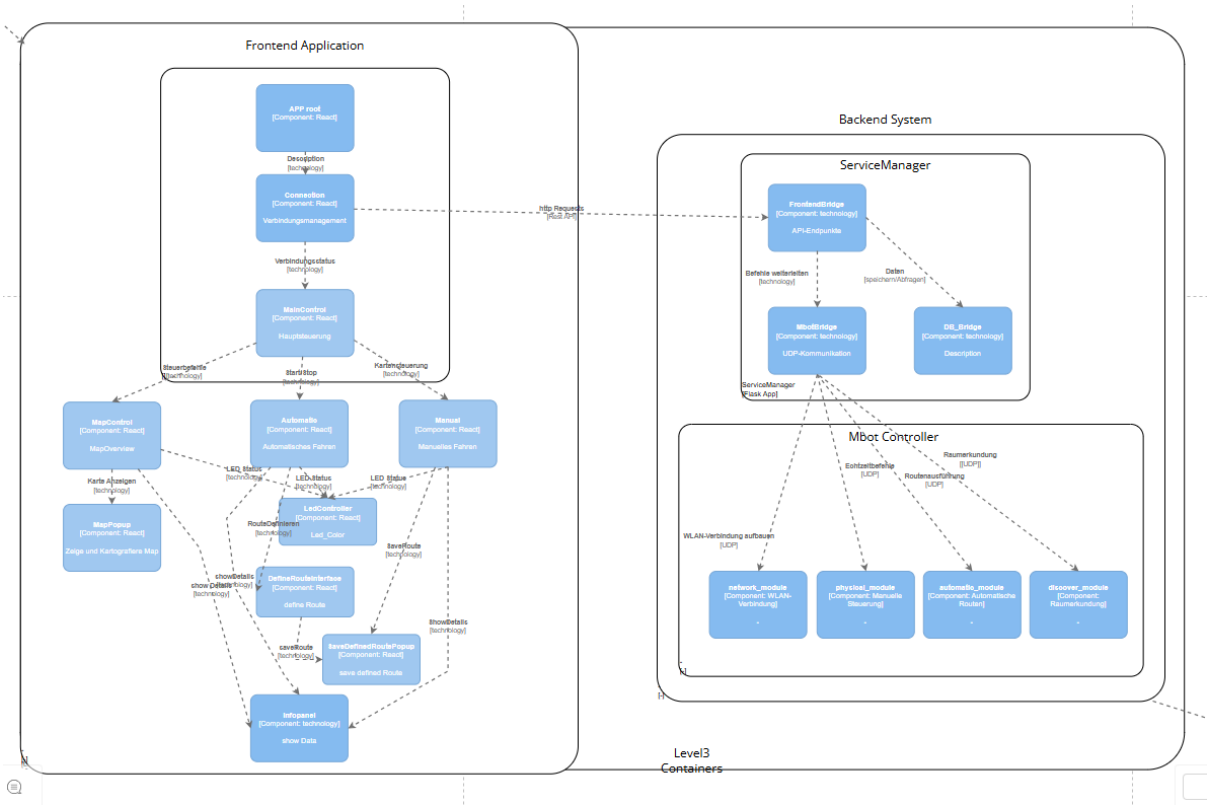
5.4 C4-Diagramm



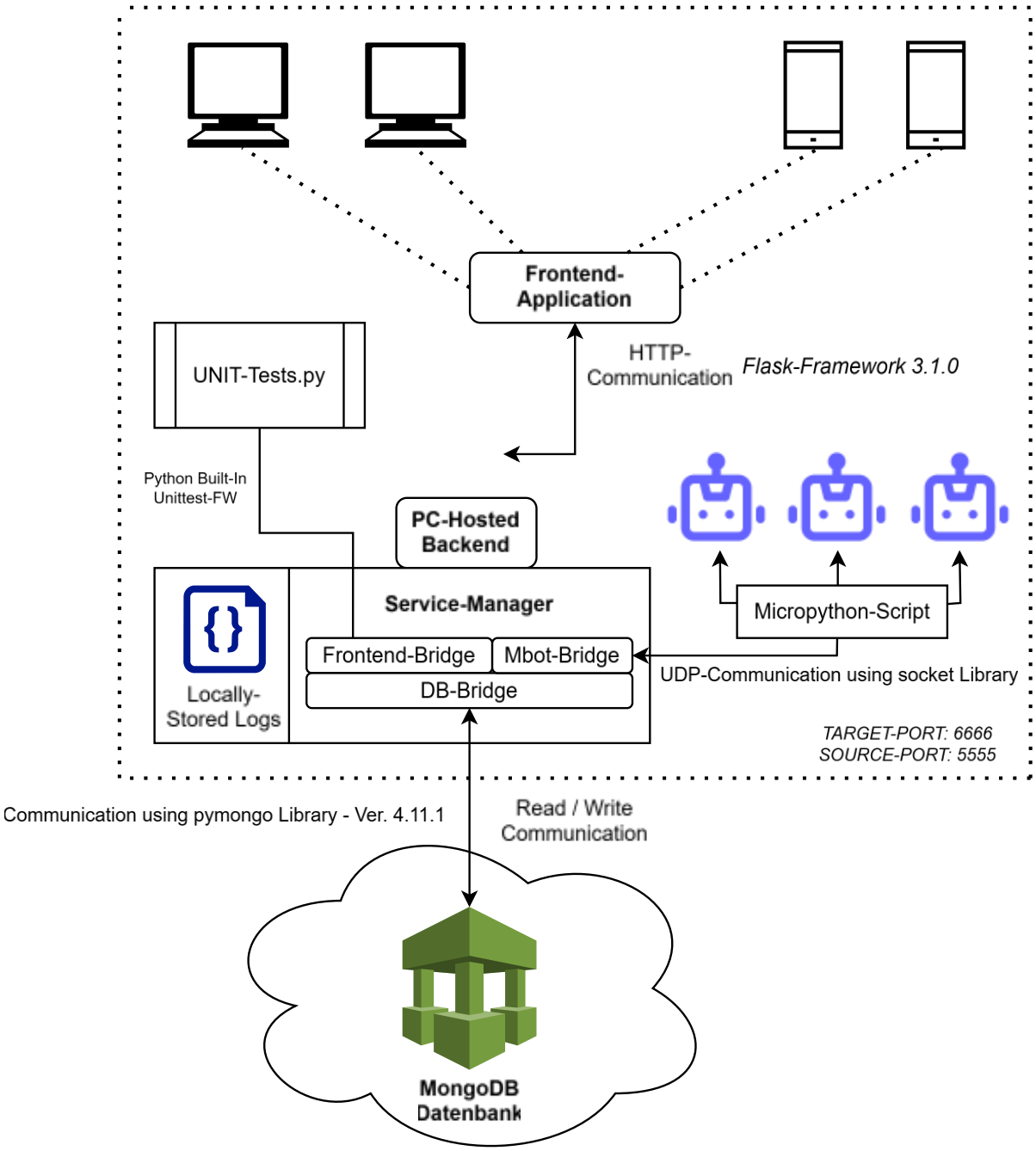
Level1
Context


User
[Person]
bedient
[HTTP / Websocket]
MBot Steuerungsystem
Web Frontend
[Container: React/Angular]
Bietet Steuerungsoberfläche und Visualiserung
API-calls
[REST / Websocket]
Backend Service
[Container: Python]
Verarbeitet Befehle, verwaltet Mbot
Steuerbefehle
[UDP]
Sensordaten
[UDP]
DB-Connection
[PyMongo]
Mbot-Script
[Container: MicroPython]
Steuert Hardwarekomponenten
DB
[Container: MongoDB]
Speichert Daten
Containers
[Level 2]
Level2
Containers
Zoom in
[-]


Backend
Backend
[Backend-Application]
FrontendBridge Module
- recording: bool
- command_log: list
- current_speed : int
- current_color: str
- anti_hons: bool
- stoproute: bool
- discovery_points: list
- connected_users: dict
- mbot_bridge: MBotBridge
- db_bridge: DB_Bridge
+ define_route(): Response
+ end_route():Response
+ receive_commands(): Response
+ logout(): Response
+ get_discover_points(): Response
uses
[-]
reads
[-]
logs commands
[-]
jsonify
uses
[-]
Mbot_bridge Module
- send_message(msg: str): None
- receive_message(): str
- configure_connection(target_ip: str, source_ip: str): None
- disconnect(): None
DB_Bridge
- push_data_DB(name: str, data: list): None
- get_route(name: str):list
FLask Framework
+ json: dict
+ remote_addr: str


Frontend Application
Backend System
ServiceManager
Mbot Controller
Level3
Containers

5.5 Verteilungsdiagramme



5.6 Softwarekomponenten / Programme

5.6.1 SW-Programme

Name	Version	Beschreibung
Visual Studio Code	1.97	IDE

5.6.2 SW-Komponenten

Name	Version	Beschreibung
Python & Python-Unittest	3.0	Hauptprogrammiersprache für das gesamte Backend
MicroPython	-	Programmiersprache für den MBot
Flask	2.0	API-Framework für das Backend
MongoDB	8.0	NoSQL-Datenbank zur Speicherung von Anwendungsdaten
ChatGPT	o4 (GPT-4.5)	KI-Unterstützung bei Code-Generierung und -Analyse
Git / Github	3.15.2	Versionskontrolle und kollaborative Codeverwaltung

6. Projektdurchführung

6.1 Sprint 1

6.1.1 Sprintplanung

Dauer: 29.01.2025 – 18.02.2925

Ausgewählte User Stories:

User-Story	Beschreibung
Netzwerkanbindung	Automatische Verbindung Schul-WLAN und IP anzeigen
Verbindung zum MBot herstellen	UDP-Verbindung einrichten und testen

Anzahl Story-Points: 105 Story-Points

Ausgewählte Punkte aus der Impediment Liste:

- Netzwerkanbindung:
  - Statusinformation mittels LED ausgeben
  - IP-Adresse anzeigen
  - Eindeutige Identifikation

- Verbindung zum MBot herstellen:
  - Kommunikation mit dem MBot über das Netzwerk
  - Unterstützung von Steuerbefehlen an den MBot
  - Antwort des MBot auf Befehle verarbeiten und ans Frontend zurücksenden

6.1.2 Sprint Demo

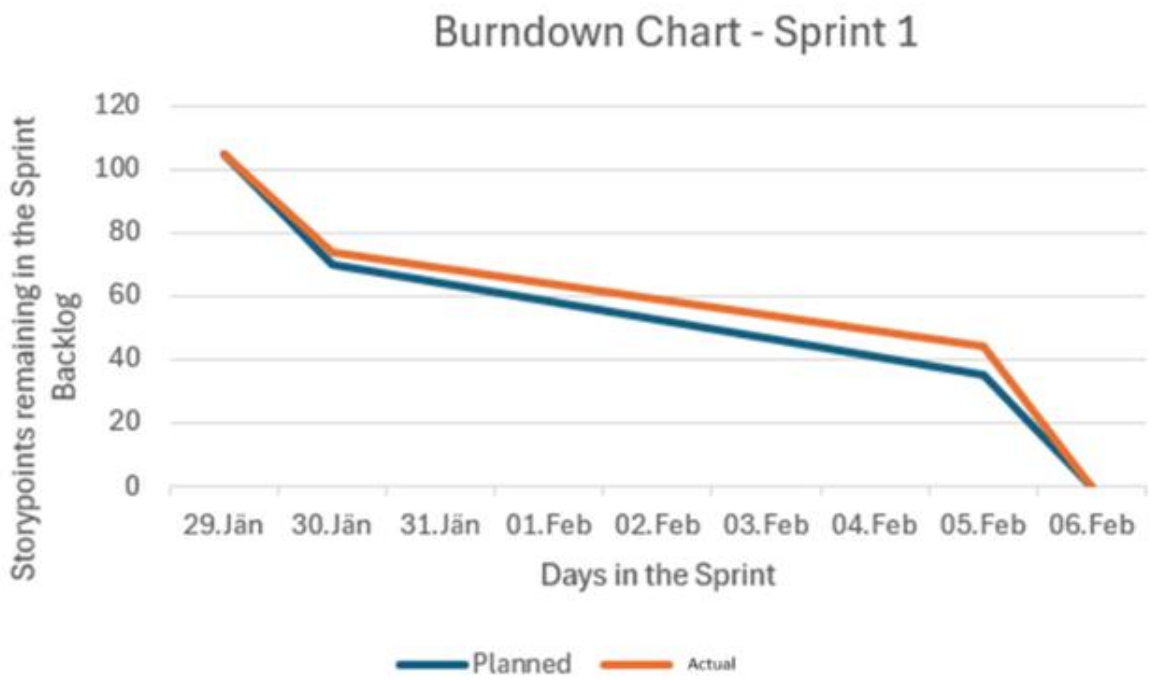
User Story Nummer	Bearbeitet von...	User Story Text	Story Points	Status
1	▪ Jonas Aberger ▪ Fabian Haslinger	Netzwerkanbindung	15	erledigt
4	▪ Jonas Aberger ▪ Tim Hechenberger	Verbindung zum Mbot herstellen	90	erledigt

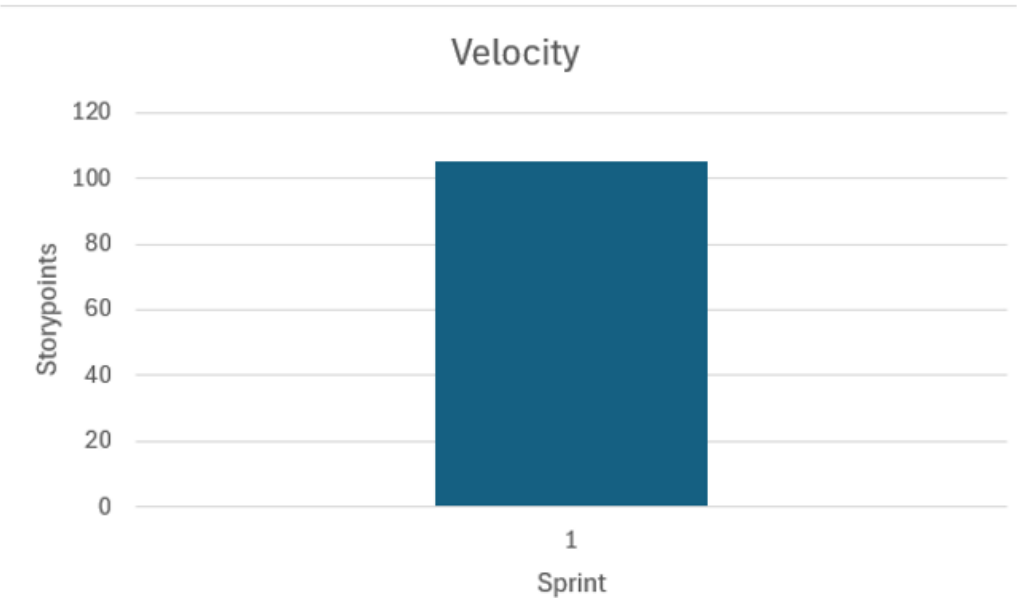
6.1.3 Sprint Retrospektive

Während des Sprints lief die Zusammenarbeit im Team sehr gut, und wir konnten die geplanten Aufgaben erfolgreich umsetzen. Anfangs brauchten wir jedoch etwas Zeit, um uns in das Projekt einzuarbeiten und die technischen Anforderungen zu verstehen.

6.1.4 Sprint Zusammenfassung

Der Sprint verlief sehr gut. Basierend auf unserer Sprint-Velocity können wir davon ausgehen, dass wir den Endtermin voraussichtlich etwas schneller als geplant erreichen werden.





6.2 Sprint 2

6.2.1 Sprintplanung

Dauer: 19.02.2025 – 12.03.2025

User-Story	**Beschreibung**
Kommunikation / Anbindung zum Frontend	Informationsaustausch zwischen Frontend & Backend mittels Schnittstellen; Entgegennahme von Befehlen
Physisches Steuerungsmodul	Ansteuerung des MBOTS; Verbindung multipler Geräte

Anzahl Story-Points: 185 Story-Points

Ausgewählte Punkte aus der Impediment Liste:

Kommunikation / Anbindung zum Frontend:

- Bereitstellung von Schnittstellen
- Genereller Informationsaustausch mit dem Frontend
- Erfolgreiche Entgegennahme von Inputs vom Frontend
- Abfertigung der Frontend-Backend Kommunikation

Physisches Steuerungsmodul:

- Erfolgreiche Anbindung an das DB-System
- Lokale Speicherung der Fahrstrecken
- Uploaden der Fahrstrecken zur DB
- Abrufen der gespeicherten Fahrstrecken

6.2.2 Sprint Demo

User Story Nummer	Bearbeitet von...	User Story Text	Story Points	Status
2	Jonas Aberger	Physisches Steuerungsmodul	85	erledigt
7	Tim Hechenberger	Kommunikation/Anbindung zum Frontend	100	Erledigt + Testing

Zusätzlich angefangene / erledigte User Stories:

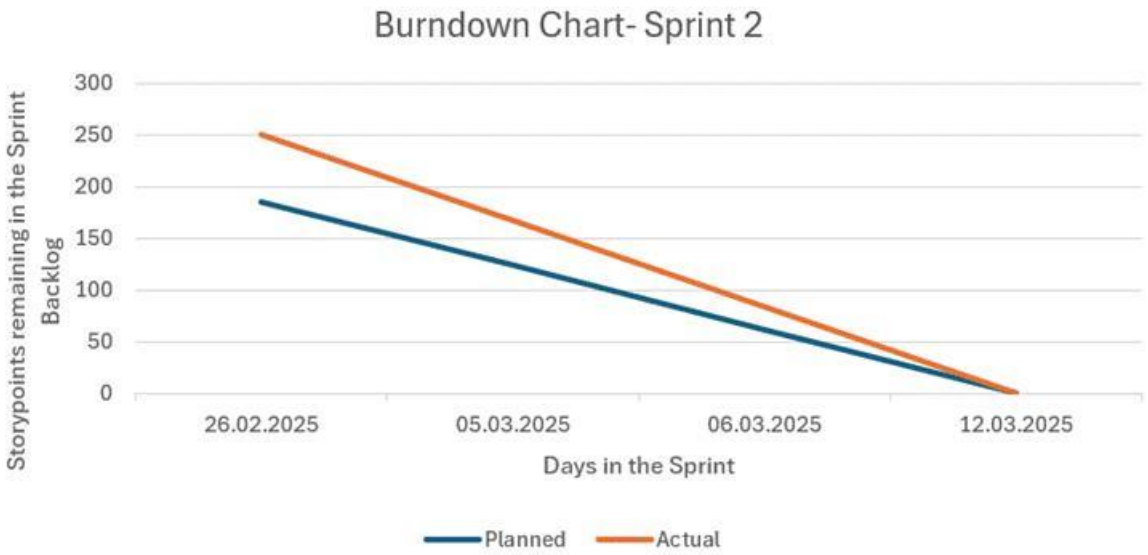
3	Jonas Aberger	Fahrstreckenlogik	80	In-progress
9	Fabian Haslinger Jonas Aberger	DB-Konfiguration DB-Kommunikation	70	erledigt

6.2.3 Sprint Retrospektive

Während des Sprints lief die Zusammenarbeit im Team sehr gut, und wir konnten die geplanten Aufgaben erfolgreich umsetzen. Da wir sehr schnell fertig wurden, konnten wir auch noch zusätzliche User Stories beginnen und finalisieren. Allerdings wurde der Fortschritt bei der Fahrstreckenlogik durch Komplikationen mit dem Frontend stark verzögert.

6.2.4 Sprint Zusammenfassung

Der Sprint verlief sehr gut. Basierend auf unserer Sprint-Velocity können wir davon ausgehen, dass wir den Endtermin voraussichtlich etwas schneller als geplant erreichen werden.



6.3 Sprint 3

6.3.1 Sprintplanung

Dauer: 13.03.2025 – 02.04.2025

User Story	Beschreibung
DB	Kommunikation und Speicherung/Laden von Fahrstrecken

Anzahl Story-Points: 65 Story-Points

Ausgewählte Punkte aus der Impediment Liste:

DB:

- Form der Datenspeicherung bestimmen
- Konfiguration & Setup der Datenbank
- Hosting der Datenbank
- Kommunikation mit der Datenbank

6.3.2 Sprint Demo

User Story Nummer	Bearbeitet von...	User Story Text	Story Points	Status
9	Fabian Haslinger Jonas Aberger	DB-Konfiguration DB-Kommunikation	65	Letzen Sprint abgeschlossen

Zusätzlich angefangene / erledigte User Stories:

3	Jonas Aberger	Fahrstreckenlogik	80	erledigt
9	Fabian Haslinger Tim Hechenberger	Kartographie-Modul	90	In Progress

Spike Story:

- **Ziel des Spikes:** Erforschung und Entwicklung eines Prototyps für den Discovery Mode des MBot, um Hindernisse zu erkennen und zu umgehen.
- **Timeboxing:** Ein fester Zeitraum wurde gesetzt, um den Prototypen zu entwickeln und Lösungen für die Hindernisvermeidung zu testen.
- **Prototypen / Variantenbildung:**
  - Verschiedene Ansätze wurden getestet, darunter:
    - *BFS (Breitensuche)*
    - *A* (A*-Algorithmus)*
    - *Neander Förmig (ein weniger strukturierter Ansatz)*
- **Lösungsansatz:** Alte Ansätze wurden verworfen, um auf einen neuen, von Herrn Eigner vorgeschlagenen Ansatz zurückzugreifen.
- **Ergebnisse:** Der Discovery Mode konnte erfolgreich implementiert werden, Hindernisse wurden korrekt erkannt und umfahren.

6.3.3 Sprint Retrospektive

Die Arbeitsatmosphäre während dieses Sprints war insgesamt positiv und von einem ruhigen, konzentrierten Miteinander geprägt. Herausforderungen traten insbesondere in der Abstimmung mit dem Frontend-Team sowie im Bereich des Discovery-Mode-Moduls auf (vgl. Spike-Story). Die Kommunikation zwischen den beteiligten Teams war teilweise unklar, was zu Missverständnissen und Verzögerungen führte.

Für den nächsten Sprint wurde beschlossen, den sprachlichen Austausch gezielter zu verbessern – etwa durch regelmäßige kurze Sync-Meetings oder klar definierte Kommunikationskanäle. Zudem soll durch die stärkere Abgrenzung und Isolation einzelner Arbeitsprozesse die Eigenverantwortung innerhalb der Teams gestärkt und Schnittstellenprobleme reduziert werden.

6.3.4 Sprint Zusammenfassung

Der Sprint verlief insgesamt erfolgreich. Es gab einige Herausforderungen im Zusammenhang mit dem Discovery Mode, die jedoch erfolgreich gelöst werden konnten. Basierend auf unserer Sprint-Velocity gehen wir davon aus, dass wir den Endtermin voraussichtlich etwas früher als ursprünglich geplant erreichen werden.

6.4 Sprint 4

6.4.1 Sprintplanung

Dauer: 30.04.2025 – 14.05.2025

User Story	Beschreibung
Discovery-Mode / Kartographie	MBot sammelt, verarbeitet und überträgt Sensordaten an das Frontend

Anzahl Story-Points: 90 Story-Points

Ausgewählte Punkte aus der Impediment Liste:

Kartographie-Modul:

- Sammeln der Daten (Sensor)
- Aufarbeitung der empfangenen Daten
- Bereitstellung & Speicherung der Daten

6.4.2 Sprint Demo

User Story Nummer	Bearbeitet von...	User Story Text	Story Points	Status
9	Tim Hechenberger Fabian Haslinger Jonas Aberger	Kartographie-Modul	90	Erledigt + Testing

Zusätzlich angefangene / erledigte User Stories:

8	Tim Hechenberger Fabian Haslinger Jonas Aberger	Dokumentation	-	In-Progress

6.4.3 Sprint Retrospektive

Die Arbeitsatmosphäre während dieses Sprints war durchweg positiv und produktiv. Die Zusammenarbeit im Team verlief reibungslos, und es traten keine größeren Herausforderungen auf. Alle Aufgaben konnten termingerecht abgeschlossen werden. Für den kommenden Sprint liegt der Fokus vor allem auf der finalen Dokumentation sowie dem Beheben kleinerer Bugs. Um den guten Workflow beizubehalten, soll die bisherige Arbeitsweise fortgeführt und bewährte Kommunikationsstrukturen beibehalten werden.

6.4.4 Sprint Zusammenfassung

Der Sprint verlief insgesamt sehr erfolgreich. Es traten keine nennenswerten Herausforderungen auf, alle Story Points konnten planmäßig abgeschlossen werden. Abgesehen von kleineren Bugfixes steht nun hauptsächlich die Dokumentation aus. Auf Basis unserer Sprint-Velocity gehen wir davon aus, den Endtermin sogar etwas früher zu erreichen.

6.5 Sprint 5

6.5.1 Sprintplanung

Dauer: 15.05.2025 – 04.06.2025

User Story	**Beschreibung**
Dokumentation	Abschluss der parallel geführten Dokumentation

Anzahl Story-Points: -

6.5.2 Sprint Retrospektive

Insgesamt war die Team-Moral im vergangenen Sprint als neutral bis ausgeglichen zu bewerten. Die finalen Aufgaben im Rahmen der Dokumentationsaufarbeitung gestalteten sich stellenweise als mühsam, konnten jedoch ohne größere Schwierigkeiten abgeschlossen werden. Während des gesamten Arbeitsprozesses traten keine nennenswerten Rückschläge auf.

6.5.3 Sprint Zusammenfassung

Die Projektdokumentation wurde auf inhaltliche Lücken überprüft und entsprechend ergänzt. Zudem wurden das MBot-Skript sowie das Hauptprogramm des Backends in Zusammenarbeit mit dem Frontend-Team umfassend auf Bugs und Fehler getestet, dabei identifizierte Probleme konnten im Anschluss – ohne weitere größere Aufwande - direkt behoben werden.

7. Installation / Software Deployment

Die fertige Softwarelösung war ursprünglich nicht für ein offizielles Deployment vorgesehen. Vorerst wird die Backend-Applikation lokal und statisch auf einem ausgewählten Rechner ausgeführt. Für eine zukünftige Nutzung und mögliche Deployment-Optionen wären zusätzliche Abstimmungen sowie Erweiterungen am Frontend erforderlich, die im Rahmen dieses Projekts jedoch weder geplant noch umgesetzt wurden.

8. Projektabschluss

8.1 Projektzusammenfassung

In diesem Projekt haben wir in fünf Sprints ein System zur Steuerung und Kommunikation mit dem MBot entwickelt. Zuerst ging es um die Verbindung mit dem Schul-WLAN und dem MBot selbst. Danach haben wir die Schnittstelle zwischen Frontend und Backend aufgebaut und den MBot so programmiert, dass er auf Steuerbefehle reagieren kann.

Im dritten Sprint wurde die Datenbank eingerichtet und der sogenannte "Discovery Mode" getestet – ein Modus, in dem der MBot Hindernisse erkennt und umfahren kann. Dabei gab es kleine Schwierigkeiten mit der Zusammenarbeit mit dem Frontend-Team, die aber gelöst wurden.
Im vierten Sprint haben wir die Sensordaten des MBots verarbeitet und an das Frontend geschickt. Alles lief reibungslos. Im letzten Sprint stand die Dokumentation im Fokus. Außerdem wurden letzte Fehler im Code behoben.

Das Projekt war insgesamt sehr erfolgreich. Alle Aufgaben konnten wie geplant oder sogar schneller abgeschlossen werden. Die Teamarbeit funktionierte gut und alle Ziele wurden erreicht.

8.2 Attachments

https://github.com/jonasaberger/SYP4_MBOT_G1 - GitHub

Datei	Nutzen
main.py	Starten des ServiceManagers
service_manager.py	Starten der API & Server
frontend_bridge.py	Stellt Verbindung zum Frontend her
db_bridge.py	Stellt Verbindung zur Datenbank her
MBot_bridge.py	Kommunikation zum MBot
UnitTests.py	Unit Tests
mblock_demo.py	File für den MBot zum Probieren
mblock_main.py	Funktionierender Code für den MBot